**Exercice 1 :**

* Afficher à l’aide d’un bloc PL/SQL le message suivant  
  « Premier exercice de PL/SQL effectué à la date du ….date du jour….. »
* Créer un programme PL/SQL qui insère une nouvelle catégorie de produit avec les valeurs suivantes : code =5 et libelle = « composants »
* Valider la transaction

**Exercice 2 :**

* Afficher en utilisant les variables les renseignements utilisateur et la date système. Le nom et le prénom de l’étudiant doivent être définis avec une constante.
* Modifier la valeur de la constante dans le corps du bloc PL/SQL
* Déclarer une nouvelle variable âge avec la contrainte not null : quand faut-il obligatoirement lui assigner une valeur ?

**Exercice 3 :**

* Utiliser les variables de substitution afin de demander l’introduction de deux nombres à l’écran. Les nombres sont récupérés dans des variables.
* Le script aura pour but d’afficher le résultat du calcul : nombre1\* nombre2
* Si le résultat est inférieur à 100, afficher le message ‘trop facile » sinon afficher ‘trop dur !’

**Exercice 4**

**Les variables scalaires**

Réaliser les deux scripts suivants (utiliser les variables de liaison)

* Demander à l’utilisateur d’introduire son nom et deux nombres au choix
* Premier script :  
  Afficher un message de bienvenue à l’utilisateur contenant son nom et les nombres introduits
* Deuxième script :  
  Si les deux nombres sont inférieurs à 100 : afficher « la somme du nombre 1…..et du nombre 2 est ……. »  
  Si le premier nombre est supérieur à 100 et le deuxième, afficher la différence entre les deux nombres.  
  Sinon afficher le produit des deux nombres.
* Réaliser des conversions implicites et des conversions explicites sur les valeurs introduites à l’écran

**Exercice 5**

Explique de façon complète (schéma + commentaires) l’utilité du PL/SQL avec Oracle (TRANSACT/SQL avec SQLSERVER) :

Qu’est-ce que le PL/SQL

Pour quelles raisons a-t-on introduit cette extension « programmation » au SQL ?

Comment intégrer ces « programmes » SQL à un langage de programmation (ex : java ?

**Exercice 6 : les blocs imbriqués**

Réaliser le bloc PLSQL suivant :

* **Travailler avec 3 blocs imbriqués**
* **Utiliser deux variables : la première est une variable de liaison qui contiendra la date système et sera définie dans le bloc principal. La deuxième variable (MaVariable sera redéfinie au niveau de chaque bloc et permettra d’afficher dans quel bloc on se situe.**

**Exercice7  
  
créer un bloc PL/SQL permettant :**

* **De mettre à jour un enregistrement de la table commande (le numéro de la cde est introduit par l’utilisateur)**
* **Si la date de la commande est nulle, introduire la date système et modifier l’état de la commande qui devient ‘traitée’**
* **Travailler avec les variables basées**
* **Afficher la commande avant la mise à jour et après la mise à jour**

**Exercice8 : instruction bulk collect et mise à jour  
  
créer un bloc PL/SQL permettant :**

* **De récupérer l’ensemble des enregistrements de la table salarié et d’afficher les renseignements concernant les salariés dont le nom est choisi par l’utilisateur.**
* **De récupérer les numéros de commandes, les taux de tva et les taux de remise et les numéros de client. Afficher ces renseignements pour une commande choisie par l’utilisateur**
* **De modifier cette dernière commande : le taux de tva passe à 21% et si le taux de remise est nul, il passe à 5%**
* **Afficher les renseignements après la modification.**

**Exercice 9**

Écrire un script permettant

* D’introduire le numéro d’un salarié
* De mettre à jour le salaire du salarié (le % d’augmentation est introduit à l’écran)
* Si la mise à jour a été effectuée de récupérer le nouveau salaire
* Travailler avec les variables basées

**Exercice 10**

On demande de modifier le script précédent pour gérer le cas d’une mise à jour non exécutée

**Exercice 11**

On demande d’écrire le script permettant de modifier la commission des vendeurs (diviser le salaire par 3) si cette commission est non nulle.

Si la modification est validée, afficher le nombre de modifications.

**Exercice 12**

Idem exercice 11 mais travailler avec les variables basées lignes

**Exercice13**

Le script a pour but d’afficher le nombre d’articles dans la table articles

**Exercice 14**

Afficher le prix hors tva par article et le prix total htva pour la quantité en stock

**Exercice 15**

Quel est le salaire moyen des employés si le type de l’attribut salaire n’est pas connu

**Exercice 16**

Afficher le code catégorie et le libellé de la catégorie d’un article choisi par l’utilisateur ;

Travailler avec des variables de type structure

**Exercice 17**

Afficher le nombre d’articles par catégorie : …. Problème ?

**Exercice 18**

Afficher le nombre d’articles différents concernant chaque commande (numéro de commande et nom du client) si ce nombre est supérieur à 2

**Exercice 19**

Ecrire un script permettant de récupérer toutes les commandes qui sont traitées.

**Exercice 20**

Ecrire un script permettant à l’utilisateur d’introduire la désignation d’un article. Le script doit permettre d’afficher les commandes relatives à cet article (numéro de commande, désignation de l’article, total à payer pour cet article)

**Exercice 21  
  
Créer un bloc PL/SQL permettant :**

* **De mettre à jour un enregistrement de la table commande (le numéro de la cde est introduit par l’utilisateur)**
* **Si la date d’envoi est nulle, introduire la date système et modifier l’état de la commande qui devient ‘traitée’**
* **Travailler avec les variables basées**
* **Afficher la commande avant la mise à jour et après la mise à jour**

**Exercice 22: instruction bulk collect et mise à jour  
  
créer un bloc PL/SQL permettant :**

* **De récupérer l’ensemble des enregistrements de la table salarié et d’afficher les renseignements concernant les salariés dont le nom est choisi par l’utilisateur.**
* **De récupérer les numéros de commandes, les taux de tva et les taux de remise et les numéros de client.**
* **De modifier les commandes : le taux de tva passe à 21% et si le taux de remise est nul, il passe à 5% après**
* **Afficher les renseignements après la modification.**

**Exercice 23**

Écrire un script permettant

* D’introduire le numéro d’un salarié
* De mettre à jour le salaire du salarié (le % d’augmentation est introduit à l’écran)
* Si la mise à jour a été effectuée de récupérer le nouveau salaire
* Travailler avec les variables basées

**Exercice 24**

On demande de modifier le script précédent pour gérer le cas d’une mise à jour non exécutée

**Exercice 25**

On demande d’écrire le script permettant de modifier la commission des vendeurs (diviser le salaire par 3) si cette commission est non nulle.

Si la modification est validée, afficher le nombre de modifications.

**Exercice26 : la clause returning**

On demande d’écrire le script permettant d’introduire un nouveau salarié et d’afficher le nom et le prénom et la fonction du nouveau salarié.

Travailler avec les variables basées (2 façons)

**Exercice 27**

On demande d’écrire le script permettant :

* Mettre à jour **toute** la table Salaries en modifiant les commissions (salaire \* pourcentage)
* Afficher les nouvelles commissions

**Exercice 28 : les curseurs**

Ecrire un bloc PL/SQL permettant d’afficher les commandes triées par rapport à la date de commande si la commande est traitée.

Afficher le nombre de commandes traitées

**Exercice 29 : les curseurs**

Afficher le nombre de commandes par client (numéro et nom) trié par rapport au nombre de commandes

**Exercice 30 : les curseurs**

Afficher la liste des clients dont le nom commence par une lettre choisie par l’utilisateur

**Exercice 31**

Créer un bloc PL/SQL qui permet d’afficher le nombre de produits commandé par client. Le pays du client sera passé en paramètre au curseur avec une valeur introduite par l’utilisateur. Utilisez la boucle for spécifique au curseur.

**Exercice32**

Créer un bloc PL/SQL qui permet de mettre à jour la date d’envoi des commandes à la date du jour et l’état de la commande à « traitée » pour les commandes dont la date d’envoi est nul. Utiliser la clause returning pour afficher les commandes modifiées.

**Exercice 33**

Ajouter 2 nouvelles commandes pour le client 17 et une pour le client 10.

Créer un bloc PL/SQL qui permet d’afficher la liste des 3 clients qui ont passé le plus grand nombre de commandes au cours des 60 derniers mois et établissez également la liste des trois membres qui ont emprunté le moins (travailler de deux façons différentes) ;

**Exercice 34**

On désire modifier le stock minimum et le stock maximum des articles en fonction du nombre de fois où l’article a été commandé.

Travailler en deux étapes

* Modification de la structure : pour chaque article, afficher le nombre d’articles commandés à ce jour (cette colonne aura la propriété invisible) (nre\_articles)
* Mise à jour : modifier la table articles en fonction du nombre d’emprunts
  + Si nbre\_articles <2 alors le stock maximum est diminué de 20%
  + Si nbre\_articles est compris entre 2 et 5, alors les stocks minimum et maximum sont multipliés par 2
  + Si nbre\_articles est >5, alors les stocks minimum et maximum sont multipliés par 3
* Travailler avec la clause returning
* Veiller à verrouiller le curseur pendant la mise à jour

**Les procédures et les fonctions stockées**

**Exercice 35**

Créer une procédure stockée permettant la mise à jour du salaire des salariés :

* Le numéro du salarié et le taux d’augmentation du salaire sont introduits par l’utilisateur
* le taux d’augmentation est inférieur à 20%
* Travailler avec la clause returning

**Exercice 36 : création et appel d’une fonction**

Créer deux fonctions stockées dans un package permettant :

* De calculer la somme et la moyenne des salaires dans la table salaries

**Exercice 37 : création et appel d’une fontion(avec arguments)**

Créer d’une fonction stockée permettant :

* D’introduire le numéro d’un salarié
* De calculer son salaire annuel en tenant compte des commissions (utiliser la fonction nvl)
* Gérer les exceptions

**Exercice 38 : création d’une fonction dans un bloc anonyme**

Créer une fonction permettant :

* De calculer le nombre de commandes par client (numéro et nom)
* De travailler avec les variables de type tableau et de type structure

**Exercice 39 : création d’une procédure dans un bloc anonyme**

Ecrire la procédure qui permet de supprimer les clients qui n’ont pas de fax ni de téléphone.

Comment pourrait-on écrire la procédure pour éviter l’inconvénient suivant :

Aucun membre ne sera supprimé si un seul membre n’est pas supprimé ?

Afficher le nombre de lignes qui ont été supprimées.

**Exercice 40 : les packages**

Créez un package nommé gestEmp avec les objets suivants :

Une fonction qui contrôle l’existence d’un salarié à partir de son numéro.

Une procédure qui permet de supprimer un salarié à partir de son numéro.

Une procédure qui permet d’augmenter le salaire d’un salarié. La procédure doit recevoir deux arguments :

* Le premier est le numéro du salarié, le deuxième le pourcentage d’augmentions.

Une procédure qui affiche tous les salariés dont le salaire est supérieur à la moyenne des salaires (cette procédure devra s’appuyer sur un curseur nommé « cur\_sal »)

**Exercice 41 : les déclencheurs**

Créer un déclencheur LMD qui se déclenche après toute instruction INSERT, UPDATE ou DELETE sur la table CLIENTS.  
Le trigger doit tracer les informations ci-dessous dans une table créée au préalable :

* La date d’exécution et le user ayant effectué l’instruction
* La valeur AVANT et APRES pour les colonnes relatives aux numéros de client, noms, adresses, code-postaux, villes et téléphones.

**Exercice 42**

On désire gérer l’ajout de cartes de fidélité pour les clients :

* On crée une table « fidelité »liée à la table clients : numéro carte, date ouverture, nombre de commandes
* Cette colonne sera mise à jour à l’aide d’un trigger à chaque nouvelle commande d’un client
* Il faut gérer la validité de la carte qui est de 1 an
* Le calcul de la remise après 10 commandes sera effectué à l’aide d’une fonction